اشهر المسائل الي تستخدموا فيها البحث ، الترتيب ، خوار...
لم مشاكل المعروفة اللي تستخدمها في الخوارزميات
(المقصود بيها مثلا اقصر مسار او العكس)
لكل نفس المشكلة لكن

الخوارزمية : هو طريقة حل لمسئلة معينة اكثر من خوارزمية
الهدف تحديد الافضل
↓
يتم تحديد الافضل على المعايير
الي من خلالها نعمل الخوارزمية
↓
يتم تحليل الخوارزمية لتقييم اداء الخوارزمية وتحديد الكفاءة
يتم تحديد الكفاءة عاملين (تعقيد الوقت ، المساحة)
زمن التنفيذ

التعابير المستخدمة في حساب زمن التنفيذ ① Big . oh (الاهم)
هو طريقة استخدام لحساب زمن التنفيذ ولحد اكبر مدة زمنية تحتاجها الخوارزمية
تم التعبير عنها بـ worst case .
(عملية المقارنة)
باش نحل الخوارزمية بشكل صحيح ← تحديد العملية
(العملية ، ما كانت) ② تم مرة تم تكرار العملية

first class
Luxury
Golden rose

باش نحسب الزمن اذ نحلل الخوارزمية عندنا طريقتين :
① تحليل النظري رياضيا المهم واللي بنركزوا عليه
② تحليل العملي

زمن التنفيذ هو الخطوات التي تستنفذ داخل الخوارزمية
worst case تاخذ الوقت الاعلى
best case تاخذ الوقت الادنى
average الحالة المتوسطة

التعريف الرياضي لـ Big oh
$f(n) = O(n)$
$f(n) = O(g(n))$ دالة تمثل تحليل زمن تنفيذ الخوارزمية

طريقة ... تحليل الخوارزمية يعني أوجد زمن التنفيذ
تعريف رياضي
اذا تحقق العلاقة $F(n) = O(g(n))$
شرطان n $c \geq 0$
$f(n) \leq C(g(n))$
يتم موجبة
صحيحة n

$c \leftarrow$ قيمة ثابتة تحدد معدل النمو
Big oh تحدد معدل الدالة

ماسود دالة تلاقي
الحد الاعلى
زمن التنفيذ يرتفع زيادة المدخلات

$f(n) = O(g(n))$
if $F(n) \leq c(g(n))$
$3n + 2 \leq c\ n$

← (لازم توجد قيمة n و c
لازم توجد n_0 اللي بتحقق العلاقة)
(n و n_0 لازم يكونوا في نقطة معينة متسلسلة)

↳ اذا كان مش موجود معيا علاقة غير صحيحة
(يعني انني محتاجين لقيمة باش تتحقق العلاقة)

* المحاضرة الثانية *

باش نتحلل رياضيا بندأ نفرض قيم لين نستنتج الحل

مثال ١:- اثبت ان الزمن التنفيذي للدالة $F(n)$ يساوي $O(n)$

يوجد طريقتان:-

$$F(n) = O(g(n))$$

$$\text{if } F(n) \leq C\,(g(n))$$

حيث ما $n_0 = 1$

$$3n + 2 \leq Cn$$

$$5 \leq C \cdot 1$$

$$n_0 = 1 \quad \leftarrow C = 5$$

طريقة الثانية:

نساوي بالمقدارين ونستنتج

$$3n + 2 \leq C\,(g(n))$$

$$3n + 2 \leq 3n + 2$$

$$3n + 3$$

$$3n + \dots$$

$$\vdots$$

$$3n + n \rightarrow$$ الدالة Big-oh

$$3n + 2 \leq 4n$$

Big-oh للدالة $O(n)$

n_0	$3n+2$	$4n$
2	$3 \times 2 + 2$ = 8	8

$C = 4$

$n_0 = 2$

مثال ②: اثبت ان زمن التنفيذ للدالة $f(n)$ يساوي $O(n^2)$

نفرض $1 = n_0$

$$F(n) = O(g(n))$$

$$\text{if } F(n) \leq C\,(g(n))$$

$$3n^2 + 10n \leq Cn^2$$

Big-oh ← $3n^2$

$$3(1)^2 + 10 * 1 \leq C \cdot 1$$

$$3 + 10 \leq C$$

$$13 \leq C$$

حيث ما $C = 13$, $n_0 = 1$

$$F(n) = O(n^2)$$

طريقة الثانية:

$$F(n) = 3n^2 + 10n$$

$$3n^2 + 10n \leq 3n^2 + 10n^2$$

$$\leq 3n^2 + 11n^2$$

$$\leq 3n^2 + 12n^2$$

$$\leq 3n^2 + n^2$$

$$\leq 4n^2$$

$C = 4$

$n_0 = 10$

" المحاضرة الثانية "

تحسبوها خطوة

لو كل عادي هاتشغل خطوة

مثال: اكتب خوارزمية لحساب مجموع الاعداد من ١ الى n

int sum = 0 ← لازم اي متغير بسند له قيمة

	time	
int sum = 0	1	→ O(1)
for(i=1 ; i≤n ; i++)	2n+2	→ O(n)
sum = sum + i ;	n	→ O(n)
return sum ;	1	→ O(1)

اني معروض بجيب زمن تنفيذ (الخطوات) كل سطر مرة

زمن التنفيذ بنطلعه بالتحليل من طريقة Big-oh

كيف بنستخدم قواعد Big oh
زمن تنفيذ كل سطر وبعده نجمعهم

كيف نعرف الخطوات، time، الي هو كم مرة تم تنفيذه

n = 6

For(i = 1 ; i≤n ; i++)

	time	
i = 1 ← كم مرة نفذناها	1	
i ≤ n (عدد مرات المقارنة) ← كم مرة تم تنفيذ هذا (المقارنة)	n+1	= 1 + (n+1) + n
		= 2n + 2
i ++ ← لانها بتتكرر	n	O(n) الحد الاعلى

في عملية المقارنة بيعملها اشي تحقق الشرط
بتركز على كم مرة تم اختبار الشرط
(العمليات الي تمت) ← اختبار الشرط

1 2 3 4 5 6 7

الزمن الكلي للخوارزمية T_n = $O(1) + O(n) + O(n) + O(1)$ (عملية جمع نأخذ الاس الاعلى) الحد
= $O(n)$ = اسوء زمن تنفيذ

وعلشان نكتب التحليل:

$F(n) = 1 + 2n+2 + n + 1$

$f(n) = 3n + 4$ ← معادلة تمثل زمن التنفيذ وتستخدم كدالة لتعريف الزمن الرياضي اذا شفنا تناظري بعض

---

n = 5   n = 6

مثال:

	time
int x = 0 ;	1
For (i = 0 ; i < n ; i++)	1 + n+1 + n = 2n+2
{	
x = x + 1 ;	n
y = x + 2 ;	n
}	
print x , y ;	1

n+1 ← n تكرار الزيادة

الفرق انها دخلت في الحلقة هذا الشي ولت n ← وهما جوى تكرار

دايما نفسه لانهم داخل الاقواس

لا تختلف في اخر سطر

i = 0
n = 6
7 = 1+6
1+n

0 1 2 3 4 5 6

$F(n) = 1 + 2n + 2 + n + n + 1$
$= 4n + 4$

Big-oh ← الزمن الكلي بدلالة $F(n)$ $O(n)$

- notation Omega Ω

$F(n) = \Omega(g(n))$ if $F(n) \geq C(g(n))$

المعامل الخامس :-

يعني Big omega يساوي بلا ... زمن التنفيذ أي أن

$F(n) = 5n^2 \Rightarrow \Omega(n^2)$

$n_0 > 0$, $n > n$

$\frac{5n^2}{n^2} \geq \frac{Cn^2}{n^2}$

$C \leq 5$

Omega → lower bound

الحد الأدنى للدالة (أقل منها أو يساويها)

theta → tight bound

$n_0 = 1$

- $F(n) = 2n+3$, $F(n) = \Omega(n)$
- $F(n) = \Omega(g(n))$

$F(n) \geq C(g(n))$

$2n+3 \geq Cn$

$2n+3 \geq Cn$

$2+3 \geq C$

$5 \geq C$

مثال :- اثبت أن زمن التنفيذ بدلالة Big omega يساوي $\Omega(n)$

$n^2 \geq Cn$

$F(n) = n^2$

$F(n) = \Omega(g(n))$

$n^2 \geq Cn$

$n^2 \geq 3 \cdot n$

$3^2 \geq 3 \cdot 3$

$9 \geq 9$ ✓ تحقق العلاقة

* notation theta θ

$F(n) = \theta(g(n)) \rightarrow O(g(n)) = \Omega(g(n))$

التعريف الرياضي

$F(n) = \theta(g(n))$ if $C_1(g(n)) \leq F(n) \leq C_2(g(n))$

$\theta(g(n)) = \Omega(g(n))$

worst case = best case

C_1 , C_2 لازم تكون ... حتى تتحقق العلاقة

ملاحظة :-

في omega دائما أفضل حالة ممكن تكون القيمة التي تأخذها كالحد الأعلى.

$F(n) = 2n^2 - 3$

O(n)	Ω(n)
O(n²)	Ω(n²)

omega أكثر حاجة تأخذها n^2 حتى لا تتحقق العلاقة

big oh الحد الأدنى n^2 وممكن تكون أعلى منها

theta تحقق لما تكون omega و big oh متساويات.

مثال :-

$F(n) = 3n+2$, $g(n) = n \Rightarrow F(n) = \theta(g(n))$

اثبت $C_1 = 3$, $C_2 = 4$

$F(n) = \theta(g(n))$ if $C_1(g(n)) \leq F(n) \leq C_2(g(n))$

$n_0 = 2$

$3n \leq 3n+2 \leq 4n$

$6 \leq 8 \leq 8$ ✓

$F(n) = \theta(n)$ $n_0 = 2$

theta θ(n)

الثابت زمن تنفيذه أقل

$1 < \log n < \sqrt{n} < n < n\log n < n^2 < n^3 < 2^n < n!$

بفرض أن O(n) ، هذا كله يعتبر upper bound وبالتالي يحققوا العلاقة

بفرض أن Ω(n) ، هذا كله يعتبر lower bound وبالتالي يحققوا الدالة

```
int x = 0;                                  1
For ( i = -2 ; i < n ; i++ )                n+3   n+2
{
    x = x + 1;                              2n+2
    y = x + 2;                              n+2
}
print x, y;                                 1
```

$F(n) = 1 + 2n + 2 + n + n + 1$

$= 4n + 4$

$T_{on} = O(n)$

```
int x = 5;                                  1
For ( i = 0 ; i <= n ; i++ )                n+2   n+1   O(n)
    For ( j = 1 ; j < n ; j++ )             n     n-1   O(n)
    {
        x = x + i + j;                      3n    O(n)
        print (x);                          n-1   O(n)
    }
```

$F(n) = 1 + 2n + 4 + n + n(n-1) + n(n-1)$

$= 1 + 2n + 4 + n^2 + n^2 - n + n^2 - n$

$F(n) = 3n^2 - n + 2 \Rightarrow O(n^2)$

```
For ( i = 1 ; i <= n ; i++ )                n+1   n   2n+2   O(n)
    sum = sum + i;                          n     O(n)
For ( j = 0 ; j <= n ; j++ )                n+2   n+1   2n+4   O(n)
{
    x = x + j;                              n+1   O(n)
}
```

$T(n) = O(n) + O(n) + O(n) + O(n) = O(n)$

$F(n) = 2n + 2 + n + 2n + 4 + n + 1 = 5n + 7$

```
For(i=1; i<n; i++)
for(j=0; j<i; j++)
{
    statment;
}
```

في حالة بداية الشرط For بـ 0 هي داخل For نفسها 1 في حالة بداية الشرط For بـ 1 الف داخل For تكون بـ n

i	j
1	0 ✓
	1 ✗
2	0 ✓
	1 ✓
	2 ✗
3	0 ✓
	1 ✓
	2 ✓
	3 ✗
n	0 ✓
	1 ✓
	2 ✓
	⋮
	n ✗

$$\sum_{j=i} = 1+2+3 \dots n$$

$$\frac{n(n+1)}{2} \Rightarrow \frac{n^2+n}{2} \Rightarrow \frac{1}{2}(n^2+n)$$

```
For(i=1; i<=n; i=i*2)        // log2 n+2
    sum = sum+i;             // log n+1
```

$n=8$

i=1	i<=n	i*2
1	✓ 1<=8	2 → 2^1
2	✓ 2<=8	4 → 2^2
4	✓ 4<=8	8 → 2^3
8	✓ 8<=8	16 → 2^4
16	✗ 16<=8	✗

2^k → $k=i$

$2^k * 2$

$2^k <= n \Rightarrow 2^k = n \Rightarrow \log_2 n$

العداد لما يكون مضروب في 2 او مقسوم يكون قيمة اللوغاريتم دائما log n

Foo) = $O(\log_2 n)$

```
For(i=n; i>=1; i=i/2)
    sum = sum+i;
```

$i=10$

i=10	i>=1	i/2
10	✓ 10>=1	5
5	✓ 5>=1	2.5
2.5	✓ 2.5>=1	1.25
1.25	✓ 1.25>=1	0.7
0.7	✗ 0.7>=1	✗

$i<=n \Rightarrow \log_2 n+1+1$

$\log_2 n+2$

$i<n \Rightarrow \log_2 n+1$

```
if(x<n)                  1 => O(1)
printf("%d",n);          1 => O(1)
else
{
    For(i=0; i<n; i++)   n+2  n+1  2n+4 => O(n)
        printf(i);       n => O(n)
}
```

$T(n) = O(n)$

شئ واضح

« في if و else ناخد الأعلى فيهم لأن بتنفذ واحده فيهم بس »

Best case : بعد ما مرتبة

نفهم نختبره بعد ما مرتبة

list اذا وبتكون مرتبة من تشتمل على

هذه الحالة تكون في حالة انها ... اختبره مرة واحدة

$$\sum_{i=2}^{n=5} t_i = 1 , 1 , 1 , 1 \quad (i=2,\ i=3,\ i=4,\ i=5)$$

هذه الحالة

$$\sum_{i=2}^{n} = t_i = 1 = 0$$

$$\sum_{i=2}^{n} = 4 \Rightarrow (n-1)$$

1	2	3	4	8

الجمل الداخلية لـ while لن يتم تنفيذها لأن الشرط غير صحيح في كل مرة .

: worst case :

عندما تكون مرتبة ترتيب عكسي

اسوء حالة تكون عندما list مرتبة ترتيب عكسي .

① $i=2 \Rightarrow t_i = 2$, ② $i=3 \Rightarrow t_i = 3$

③ $i=4 \Rightarrow t_i = 4$, ④ $i=n = t_i = n$

7	5	4	3	1

while

$$\therefore \sum_{i=2}^{n} t_i = 2+3+4+\dots n$$

$$\frac{n(n+1)}{2} - 1 \rightarrow$$ لأن تبدأ بـ 2

مثال للتوضيح

$$\sum_{i=1}^{n} i = 1+2+\dots n = \frac{n(n+1)}{2}$$

statement	best case	worst case
For i=2 to n	n	n
A[i] = key	n-1	n-1
j = i-1	n-1	n-1
while (j>0) & (A[j]>key)	n-1	$\frac{n(n+1)}{2} - 1$
A[j+1] = A[j]	0	$\frac{n(n-1)}{2}$
j = j+1	0	$\frac{n(n-1)}{2}$
A[j+1] = key	n-1	n-1
	$5n-4 \Rightarrow \Omega(n)$	$\frac{7x+3x^2}{2} - 4 \Rightarrow O(n^2)$

ملاحظة :-

في حالة A[j+1] = A[j]

$$\sum_{i}^{n} t_i = 1+2+3\dots(n-1) \rightarrow \frac{n(n-1)}{2}$$

- Ex:- $F(n) = 10n^4 + 3n^2 + 5n$

T 1- $T(n) = O(n^5)$ $n^4 \leq n^5$ ✓

F 2- $T(n) = O(n \log n)$ $n^4 \not\leq n \log n$ ×

T 3- $T(n) = O(n^7)$ $n^4 \leq n^2$ ✓

F 4- $T(n) = \Omega(n^5)$ $n^4 \geq n^5$ ×

T 5- $T(n) = \Omega(n^3)$ $n^4 \geq n^3$ ✓

F 6- $T(n) = \Theta(n^2)$ $n^4 = n^2$ ×

T 7- $T(n) = \Theta(n^4)$ $n^4 = n^4$ ✓

- اثبت هاذا كانت العلاقات صحيحة

$F(n) \leq C(g(n))$

$n^4 \leq n \log n$ ×

1- $F(n) \leq C(g(n))$ O
2- $F(n) \geq C(g(n))$ Ω
3- $C_1(g(n)) \leq F(n) \leq C_2(g(n))$

- إيجاد العلاقات باستخدام النهايات:

$\lim_{n\to\infty} \frac{F(n)}{g(n)}$

1. $0 \Rightarrow F(n) = O(g(n))$
2. $\infty \Rightarrow F(n) = \Omega(g(n))$
3. $C \Rightarrow F(n) = \Theta(g(n))$

مثال:

$F(n) = 5n^2 - 4n - 10$ ، $g(n) = n^2$

اثبت هاذا كانت O أو Ω أو Θ بإستخدام النهايات.

$$\lim_{n\to\infty} \frac{F(n)}{g(n)} \to \lim_{n\to\infty} \frac{5n^2 - 4n - 10}{n^2} \to \lim_{n\to\infty} \frac{5n^2}{n^2} - \frac{4n}{n^2} - \frac{10}{n^2}$$

$$\lim_{n\to\infty} 5 - \frac{4}{n} - \frac{10}{n} \to 5 - \frac{4}{\infty} - \frac{10}{\infty} = 5 \Rightarrow F(n) = \Theta(g(n))$$

مثال:

$F(n) = n^2$ ، $g(n) = n \log n$

$$\lim_{n\to\infty} \frac{F(n)}{g(n)} \to \frac{n^2}{n \log n} \to \lim_{n\to\infty} \frac{n}{\log n} \to \frac{\infty}{\log \infty} = \infty$$

$F(n) = \Omega(g(n))$

مثال:

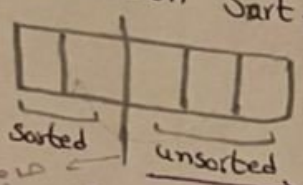
$F(n) = \sqrt[3]{n}$ ، $g(n) = \sqrt{n}$

$$\lim_{n\to\infty} \frac{F(n)}{g(n)} \to \lim_{n\to\infty} \frac{\sqrt[3]{n}}{\sqrt{n}} \to \lim_{n\to\infty} \frac{n^{1/3}}{n^{1/2}}$$

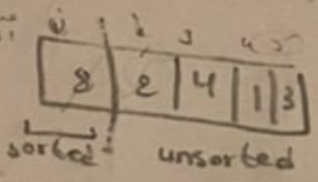
$$\lim_{n\to\infty} \frac{n^{1/3}}{n^{1/2}} \to \lim_{n\to\infty} n^{1/3 - 1/2} \to \lim_{n\to\infty} n^{-1/6} \to \lim_{n\to\infty} \frac{1}{n^{1/6}}$$

$\frac{1}{\infty} \to 0 \to F(n) = O(g(n))$

Insertion Sort



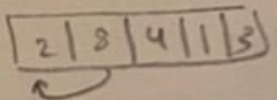
مثال:

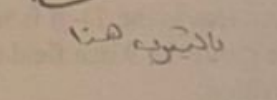



```
Insertion Sort (A, n)
{
For i = 2 to n {
    key = A[i]
    j = i - 1
    while (j > 0) and (A[j] > key)
    {
        A[j+1] = A[j]
        j = j - 1
    }
    A[j+1] = key
}
```

$O(n)$

$O(n)^2$





statement	time
For i=2 to n	n
key = A[i]	$n-1$
j = i - 1	$n-1$
while (j>0) and (A[j] > key)	$\sum_{i=2}^{n} t_i$
A[j+1] = A[j]	$\sum_{i=2}^{n} t_i - 1$
j = j - 1	$\sum_{i=2}^{n} t_i - 1$
A[j+1] = key	$n-1$

خوارزميات 7 No: Date:

Non Reusive Algortime

Algorithm

Iteration
non Reusion
Algorithm

Recursion التكرار المستمر
Algoritthm
عملية تقيد نفسها
في البرمجة هي عبارة عن
طريقة لدالة تستدعي نفسها

Function2 { جملة شرطية
للتوقف
عملية الاستدعاء
عملية الاستدعاء

Function 2()

Recursion function

① Recursion base gernal case الاستدعاء الذاتي للدالة
② bace case نقطة توقف اكوان (إيقاف الاستدعاء)

Factorial (x) = X * Factoral (X-1)

مثال: خوارزمية لحساب مضروب العدد n!
n=5
5! = 5 * 4 * 3 * 2 * 1
قيمتها
120 Factorial (5) = 5 * Factoral (5-1) = 4
24 Factorial (4) = 4 * Factoral (4-1) = 3
6 Factorial (3) = 3 * Factoral (3-1) = 2
2 Factorial (2) = 2 * Factorial (2+1) = 1
Factorial (1) = 1 ← bace case

NOTEBOOK

No: Date:

الفرق بين A, B :-

Resurion Algorathim

الان فيه دالة الاستدعاء الذاتي

non Resurion

خوارزمية لا تحتوي على دالة الاستدعاء ، تحتوي على جمل التكرار For

باش نحسب زمن التنفيذ B لازم نجيب زمن تنفيذ كل سطر اما A نكون

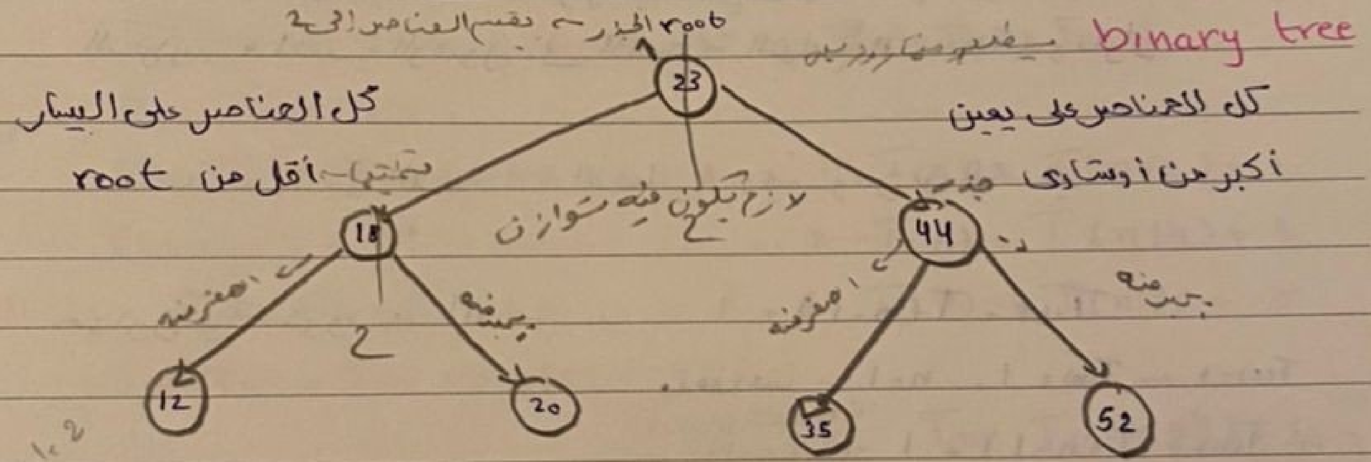
طول $T(n) = T(n-1) + 1$

يشير الى أول عنصر lo — mid — hi ← يشير الى آخر عنصر

lo			mid			hi
12	18	20	23	35	44	52

خوارزمية البحث

binary tree



if (lo > hi) ← مافيش عناصر

return false

mid [(lo + hi)/2]

في هذه الخوارزمية تقوم بتقسيم العناصر الموجودة في القائمة الى جزئين بتقسيم عدد العناصر على 2 . وتقوم بوضع العنصر الاوسط في الجذر ، والعناصر في يمين الجذر أكبر منه والعناصر يسار الجذر اصغر منه . ثم تقوم برؤية العنصر المراد البحث عنه ان كان يساوي الجذر ترجع قيمته من اول خطوة وإن لا تقوم برؤية هل هو اصغر ام أكبر من الجذر ، ان كان اكبر تذهب للعناصر يمين الجذر وإن كان اصغر نبحث الى العناصر يسار الجذر

① Recursion ← $\frac{n}{2}$

② base case = C ← عدد شروط الخوارزمية

$T(n) = T(n/2) + 1$

$4 \leftarrow 3.5 \leftarrow (7/2) \leftarrow [(lo + hi)/2] \leftarrow mid$

hi			mid			lo
12	18	20	23	35	44	52
a_0	a_1	a_2	a_3	a_4	a_5	a_6

No: Date:

يوجد طريقين A, B

Algoratim A "Rasursion"

```
int Factorial (n) {
    if (n=1)
      return 1
    else
      return n * Factorial(n-1)
}
```

$T(n)$ = [زمن دالة الاستدعاء] + [ ]

Recurion Base case

الزمن المستغرق لحساب n عن طريق العبارة

$T(n) = T(n-1) + 1$

معادلة التكرار التي تصف زمن التنفيذ

$T(5) = T(5) + 1$

$T(4) = T(4) + 1$

$T(5) = T(5) + 1 \Rightarrow T(4) = T(3) + 1$

نكتب 1 بدل كتابة 1+1+1

Algoritm B "non Recursion"

```
{ int factorila = 1, n ;          O(1)
  For (i=1; i<=n; i++)            O(n)
  Factorial = Factoril *(i+1)     O(n)
}
```

معادلة تصف حساب المضروب

i	fact	fact * (i+1)	time
0	1	1* (0+1)	T(n)
1	1	1* (1+1)	O(1) + O(n)
2	2	2* (2+1)	+ O(n)
3	6	6* (3+1)	= O(n)
4	24	24*(4+1)	
5	120		

بدينا من الصفر بأن يطلع واحد في البداية

NOTEBOOK

Insertion Sort

A	8	2	4	1	3

i = 2

① while (1 > 0 & 8 > 2) ✓

A [j+1] = A (j) .

j = 0

	8	8	4	1	3

② while (0 > 0 &)

	2	8	4	1	3

key = 4

③ w (2 > 0 & 8 > 4) ✓

	2	8	8	1	3

④ w (1 > 0 & 2 > 4) ×

	2	4	8	1	3

key = 1

w (8 > 0 & 8 > 1) ✓

	2	4	8	8	3

w (2 > 0 & 4 > 1) ✓

	2	4	4	8	3

w (1 > 0 & 2 > 1)

j = 0

	2	2	4	8	3

w (0 > 0) ×

	1	2	4	8	3

w (4 > 0 & 8 > 3) ✓

	1	2	4	8	8

w (3 > 0 & 4 > 3) ✓

	1	2	4	4	8

w (2 > 0 & 2 > 3) ×

	1	2	3	4	8

while	statment	time
i = 2	For i = 2 , 0 n	n
w = 2	key = A [i]	n - 1
key = 2	J = i - 1	n - 1
j = 1 0		
i = 3	while (j > 0 & A[j] > key	$\sum_{i=2}^{n} t_i$
key = 4	A [j+1] = A(j)	$\sum_{i=2}^{n} t_i - 1$
j = 2 1		
i = 4		
key = 1	j = j - 1	$\sum_{i=2}^{n} t_i - 1$
J = 3 2 1 0		
i = 5		
key = 3	A (j+1] = key	n - 1
J = 4 2 2		

best case

$$\sum_{i}^{n} t_i = (n-)$$

$$\sum_{i=2} t_i - 1 = 0$$

No: Date: "محاضرة 3"

Iteration method أول طريقة لتحليل معادلات التكرار

Ex:- $T(n) = T(n-1) + 1$

$T(n) = 1$

$T(1) = 1$

* تحليل الجمل التكرارية:

لازم نحاول الجمل التكرارية للوصول لمعادلة بدلالة Big oh.

يوجد 3 طرق لحل معادلات التكرار

أول طريقة Iteration method

وهي التعويض في الجملة التكرارية الأولى والثانية إلى n من الجمل التكرارية لغاية الوصول إلى الصيغة النهائية للمعادلة بدون تكرار

حسنوا من المعادلة الأصلية وليبدوا من التفير

① حوار رياضية مضروب !n ← $T(n) = T(n-1) + 1$

وتقف عند ← $n=1$ ← $T(n)=1$ ← $T(1)=1$

① $T(n) = T(n-1) + 1$ الآن رح تعيد T(n) ← $T(n-1)$!!

$T(n-1) = T(n-1-1) + 1$

$= T(n-2) + 1$

② $T(n) = T(n-2) + 1 + 1$

$T(n) = T(n-2) + 2$

$T(n-2) = T(n-2-1) + 1$

$T(n-3) + 1$

③ $T(n) = T(n-3) + 1 + 2$

$T(n-3) + 3$ نضيف k

1- $T(n) = T(n-1) + 1$

2- $T(n) = T(n-2) + 2$

3- $T(n) = T(n-3) + 3$

k- $T(n) = T(n-k) + k$

↳ الصيغة العامة

من هذه التكرارات نقدروا نستنجوا الصيغة العامة بدلالة k.

نقدر نستنتج الصيغة العامة من بعد 3 تكرارات بس أكثر

بحيث نقدر نوقف التكرار

$x = a^{x} \Rightarrow y = \log_a x$

$k = \log_2 n$

No: Date:

باخذ الشرط والتعويض في الصيغة العامة

$= k) + k$

بالتعويض في الصيغة العامة. $\rightarrow$ $\boxed{k = n-1}$

$+ n-1$

$+ n-1$

$\boxed{T(n) = n}$ ← الصيغة النهائية

$\therefore T(n) = O(n)$

---

② في مثال خوارزمية Binary search tree

$T(n) = T(n/2) + 1 \Leftarrow n = 1 \Leftarrow T(n) = 1 \Leftarrow T(n) = 1 \Leftarrow T(n) \Leftarrow$ الشرط

① $T(n) = T(n/2) + 1$

$T(n/2) = T(n/4) + 1$

② $T(n) = T(n/4) + 1 + 1$

$= T(n/4) + 2$

$T(n/4) = T(n/8) + 1$

③ $T(n) = T(n/8) + 1 + 2$

$T(n/8) + 3$

1- $T(n) = T(n/2) + 1$

2- $T(n) = T(n/4) + 2$

3- $T(n) = T(n/8) + 3$

k. $T(n) = T(n/2^k) + k$ ← الصيغة العامة

نقطة التوقف $T(1) = 1$ $T(n) = 1$ ونعوض بهما في الصيغة العامة

$T(n) = T(n/2^k) + k \Rightarrow n/2^k = 1 \Rightarrow n = 2^k$

$k = \log n$

$T(n) = T(1) + \log_2 n \Rightarrow T(n) = \log n + 1$

$\therefore T(n) = O(\log n)$

No: Date:

باخذ الشرط والتعويض في الصيغة العامة $T(1) \leftarrow T(n)=1$

$T(n) = T(n-k) + k$ ← الصيغة العامة

ديما بتعوض في قيمة الوقف

بالتعويض في الصيغة العامة. $n-k=1 \rightarrow \boxed{k=n-1}$

عوضنا في واحد

$T(n) = T(1) + n-1$

$T(n) = 1 + n-1$

$\boxed{T(n) = n}$ ← الصيغة النهائية

$\therefore T(n) = O(n)$

---

② في مثال خوارزمية Binary search tree

$T(n) = T(n/2) + 1 \Leftarrow n=1 \leftarrow T(n)=1 \leftarrow T(n)=1 \leftarrow T(n)=1$ ← الشرط

① $T(n) = T(n/2) + 1$

$T(n/2) = T(n/4) + 1$

② $T(n) = T(n/4) + 1 + 1$

$= T(n/4) + 2$

$T(n/4) = T(n/8) + 1$

③ $T(n) = T(n/8) + 1 + 2$

$T(n/8) + 3$

1- $T(n) = T(n/2) + 1$

2- $T(n) = T(n/4) + 2$

3- $T(n) = T(n/8) + 3$

k. $T(n) = T(n/2^k) + k$ ← الصيغة العامة

نقطة التوقف $T(n)=1$ $T(1)=1$ ونعوض ابيها في الصيغة العامة

$T(n) = T(n/2^k) + k \Rightarrow n/2^k = 1 \Rightarrow n = 2^k$

$k = \log n$

$T(n) = T(1) + \log_2 n \Rightarrow T(n) = \log n + 1$

$\therefore T(n) = O(\log n)$

No: Date:

مثال (3)

$T(n) = 2T(n-1)+1$, $n=0 \rightarrow T(n)=0$

① $T(n) = 2T(n-1)+1$

$T(n-1) = 2T(n-1-1)+1$

$2T(n-2)+1$

② $T(n) = 2[2T(n-2)+1]+1$

$4T(n-2)+2+1$

$4T(n-2)+3$

$(n-2) = 2T(n-2-1)+1 \Longrightarrow 2T(n-3)+1$

③ $T(n) = 4[2T(n-3)+1]+3$

$8T(n-3)+4+3$

$8T(n-3)+7$

1- $T(n) = 2T(n-1)+1$

2- $T(n) = 4T(n-2)+3$

3- $T(n) = 8T(n-3)+7$

k- $T(n) = 2^k T(n-k)+2^k-1$

$2^1 \cdot 2$

2^2 4

2^3 8

$T(n) = 2^k T(n-k)+(2^k-1)$ $\Rightarrow$ الصيغة العامة

بتعويض $n=0 \leftarrow T(n)=0$

$T(n) = 2^k T(0)+2^k-1$

$= 2^k(0)+2^k-1$

$= 2^k-1$

بما أن $n-k=0 \leftarrow n=k$ بالتعويض في

$T(n) = 2^n-1$

$\therefore T(n) = \mathcal{O}(2^n)$

NOTEBOOK